

DATABASE

TRENDS AND APPLICATIONS

Solutions for the Information Project Team • www.dbta.com

Volume 20, Number 10 • October 2007

APPLICATIONS INSIGHT



GUY HARRISON

Application Development 2007 Gets More Agile

Ever since software applications became significant business assets, there have been complaints about development projects running late, delivering the wrong result or being poor quality. It had long been hoped that as software engineering matured, it would approach the predictability and consistency of other engineering disciplines. Yet software development in practice seems unable to overcome key management issues. In *The Mythical Man Month* (1975), Fred Brooks first coined the rule “Adding manpower to a late software project makes it later.” But even now, 30 years later, it is still commonplace to see overstaffing of late projects.

Most practitioners would agree that some progress was achieved by abandoning monolithic “waterfall” methodologies in favor of iterative and agile techniques. In a waterfall methodology, all analysis and design occurred at the start of a long cycle, all testing at the end, and all construction in the middle. Requirements changes during the project were incredibly difficult and almost inevitable, since stakeholders often do not understand their true requirements

until faced with a working system.

Today, applications are typically constructed in multiple iterations, each of which attempt to implement only a single aspect of a system. This “divide and conquer” technique makes tracking and course correction more feasible.

The Agile Manifesto (<http://agilemanifesto.org>) was introduced in 2001 to describe a general set of approaches believed to be more conducive to effective software development. Agile techniques favor feedback, collaboration and course correction over specification, contract and planning. Agile techniques are often paired with another best practice - test-driven development - in which automated tests are constructed for each software unit before the unit is programmed. Today’s agile darling - SCRUM - is based on the principles of empirical process control. Software development is too chaotic, dynamic and complex to fully plan and estimate in advance. Instead, SCRUM subjects the project to a tight feedback and measurement loop. Reliable completion dates in SCRUM can only be determined after the development has been underway for

at least a few of these 30-day iterations after which one has collected enough empirical data to rationally estimate a completion date.

At the end of the day, it’s worth reminding ourselves of a few truths that are often forgotten in the search for the ultimate software development methodology: Software development is one of the most intellectually challenging activities ever introduced into business life; it’s not at all surprising that it seems so prone to failure and disappointment. And second, methodologies often promise revolutionary improvements, but nothing works as well as hiring talented software developers.

Guy Harrison is a chief architect for database solutions at Quest Software, and is a recognized expert with over 15 years experience in application and database administration, development, performance tuning and project management. Harrison is the author of Oracle SQL High Performance Tuning (Prentice Hall) and MySQL Stored Procedure Programming (O’Reilly), and is a regular speaker at trade shows and events.